Quasispecies Data

Gregori, J and Guerrero, M

09/05/2018

Contents

1 Intro

The viral quasispecies is currently defined as a collection of closely related viral genomes undergoing a continuous process of genetic variation, competition between the variants generated, and selection of the fittest genomes in a given environment. (E. Domingo, Sheldon, and Perales 2012)

The high replication error rate that generates this quasispecies is due to a lack of genetic proofreading mechanisms, and it is estimated that for viruses with typically high replicative rates, every possible point mutation and many double mutations are generated with each viral replication cycle, and these may be present within the population at any time.

Quasispecies complexity can explain or predict the behavior of a virus; hence, it has an obvious interest for clinical reasons. We are often interested in comparing the viral diversity indices between sequential samples from a single patient or between samples from different groups of patients. These comparisons can provide information on the patient's clinical progression or the appropriateness of a given treatment. (Gregori et al. 2016) (Gregori et al. 2014)

QSUtils is a package intended for use with quasispecies amplicon data obtained by NGS, but it could also be useful for analyzing 16S/18S ribosomal-based metagenomics or tumor genetic diversity by amplicons.

In this tutorial, we illustrate use of the functions provided in the package to explore and manipulate sequence alignments, convert reads to haplotypes and frequencies, repair reads, intersect strand haplotypes, and visualize haplotype alignments.

2 Install package

```r
library(devtools)
library(Biostrings)
library(ape)
library(ggplot2)

install_git("https://github.com/VHIRHepatiques/QSutils")
library(QSutils)
```

3 Data

The package contains functions that work on quasispecies data, defined by an alignment of haplotypes and their frequencies. Data are loaded from fasta formatted files, where the header of each sequence describes the ID of a haplotype and its corresponding frequency in the quasispecies population. These two pieces of information are separated by a vertical bar '|'. When frequency information is missing, each sequence is considered as a single read.

```
cat(readLines("../inst/extdata/ToyData_10_50_1000.fna") , sep = "\n")
## >Hpl_0_0001|464|46.4
## CACCCTGTGACCAGTGTGTGCGAGTTTGGTGCCCCGTTGGCATTGACTAT
## >Hpl_1_0001|62|6.2
## CACTCTGTGACCAGTGTGTGCGAGTTTGGTGCCCCGTTGGCATTGACTAT
## >Hpl_1_0002|39|3.9
## CACCCTGTGACCAGCGTGTGCGAGTTTGGTGCCCCGTTGGCATTGACTAT
## >Hpl_1_0003|27|2.7
## CACCCTGTGACCAGTGTGTGCGAGTTTGGTGCCCCGTTGGCATTGACTAC
## >Hpl_2_0001|37|3.7
## CACTCTGTGACCAGTGTGTGCGAGTTTGGTGCCCCGTTAGCATTGACTAT
## >Hpl_4_0001|16|1.6
## CACTCGGTGACCAGTGTGCGTGAGTTTGGTGCCCCGTTGGCATTGACTAT
## >Hpl_5_0001|33|3.3
## CACTCTGTGACCAGTGTGCGCGAGTTTAGTGCCCTGTCGGCATTGACTAT
## >Hpl_8_0001|54|5.4
## CACTCTGTGATCAGTGTGCGCGAGTTTAGTGCCCTGCCGGCATCGACTAT
## >Hpl_9_0001|248|24.8
## CACTCTGTGATCAGTGTGCGCGAGTTTAGTGCCCTGCCGGCATCGACTAC
## >Hpl_10_0001|20|2
## CACTCTGTGATCAGTGTGCGCGAGTTTAGTGCCCTGCCGGCACCGACTAC
```

The `ReadAmplSeqs` function loads the data from the fasta file and returns a list with two elements: the DNAStringSet `hseqs` with haplotype sequences, and the vector of counts, `nr`.

```
lst <- ReadAmplSeqs("../inst/extdata/ToyData_10_50_1000.fna",type="DNA")
lst
## $nr
##  [1] 464  62  39  27  37  16  33  54 248  20
##
## $hseqs
##   A DNAStringSet instance of length 10
##      width seq                                        names
##  [1]    50 CACCCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_0_0001|464|46.4
##  [2]    50 CACTCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_1_0001|62|6.2
##  [3]    50 CACCCTGTGACCAGCGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_1_0002|39|3.9
##  [4]    50 CACCCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAC Hpl_1_0003|27|2.7
##  [5]    50 CACTCTGTGACCAGTGTGTGC...TGCCCCGTTAGCATTGACTAT Hpl_2_0001|37|3.7
##  [6]    50 CACTCGGTGACCAGTGTGCGT...TGCCCCGTTGGCATTGACTAT Hpl_4_0001|16|1.6
##  [7]    50 CACTCTGTGACCAGTGTGCGC...TGCCCTGTCGGCATTGACTAT Hpl_5_0001|33|3.3
##  [8]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCATCGACTAT Hpl_8_0001|54|5.4
##  [9]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCATCGACTAC Hpl_9_0001|248|24.8
## [10]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCACCGACTAC Hpl_10_0001|20|2
```

The `GetQSData` function loads the data from the fasta file, removes haplotypes with relative abundances below a given threshold, and sorts the remaining haplotypes, first by an increasing number of mutations with respect to the dominant haplotype and then by decreasing frequencies within the number of mutations.

```
lstG <- GetQSData("../inst/extdata/ToyData_10_50_1000.fna",min.pct= 2,
        type="DNA")
lstG
```

```
## $seqs
##   A DNAStringSet instance of length 9
##     width seq                                               names
## [1]    50 CACCCTGTGACCAGTGTGTGCG...TGCCCCGTTGGCATTGACTAT Hpl_0_0001
## [2]    50 CACTCTGTGACCAGTGTGTGCG...TGCCCCGTTGGCATTGACTAT Hpl_1_0001
## [3]    50 CACCCTGTGACCAGCGTGTGCG...TGCCCCGTTGGCATTGACTAT Hpl_1_0002
## [4]    50 CACCCTGTGACCAGTGTGTGCG...TGCCCCGTTGGCATTGACTAC Hpl_1_0003
## [5]    50 CACTCTGTGACCAGTGTGTGCG...TGCCCCGTTAGCATTGACTAT Hpl_2_0001
## [6]    50 CACTCTGTGACCAGTGTGCGCG...TGCCCTGTCGGCATTGACTAT Hpl_5_0001
## [7]    50 CACTCTGTGATCAGTGTGCGCG...TGCCCTGCCGGCATCGACTAT Hpl_8_0001
## [8]    50 CACTCTGTGATCAGTGTGCGCG...TGCCCTGCCGGCATCGACTAC Hpl_9_0001
## [9]    50 CACTCTGTGATCAGTGTGCGCG...TGCCCTGCCGGCACCGACTAC Hpl_10_0001
##
## $nr
## [1] 464  62  39  27  37  33  54 248  20
##
## $nm
## [1]  0  1  1  1  2  5  8  9 10
```

Note that the haplotype present in an amount below 2% of the total population has now been removed.

3.0.1 Collapsing reads to haplotypes

Although `ReadAmplSeqs` can read fasta files that have no frequency information, it is more efficient to use `Biostrings::readDNAStringSet` directly. Then reads can be converted to haplotypes and frequencies with the help of function `Collapse`.

```
reads <- readDNAStringSet("../inst/extdata/Toy.GapsAndNs.fna")
reads
##   A DNAStringSet instance of length 100
##       width seq                                             names
##   [1]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGNCGC
##   [2]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
##   [3]    50 TGACGCGCACAGAGTGCTGCT...GGNTACCCCGTCGTGGTCGC
##   [4]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
##   [5]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
##   ...   ... ...
##  [96]    50 TGACGCNCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
##  [97]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
##  [98]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
##  [99]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
## [100]    50 TGACGCGCACAGAGTGCTGCT...GGTTACCCCGTCGTGGTCGC
```

```
lstCollapsed <- Collapse(reads)
str <- DottedAlignment(lstCollapsed$hseqs)
data.frame(Hpl=str,nr=lstCollapsed$nr)
##                                                   Hpl nr
## 1  TGACGCGCACAGAGTGCTGCTAAATGACTGGGTTACCCCGTCGTGGTCGC 61
## 2  ......-..........................................  2
## 3  ........................-.........................  2
## 4  ........................................-.........  2
## 5  ..........................................-.......  2
## 6  .....................-............................  2
## 7  ......N...........................................  2
## 8  -.......................................-.........  1
## 9  .-................................................  1
## 10 ....-.......-................-....................  1
## 11 ...............-..................................  1
## 12 ........................N...........-.....N.......  1
## 13 ................................N.................  1
## 14 ..............................................N...  1
## 15 ................................................-.  1
## 16 ................................................N.  1
## 17 ..........................................N.......  1
## 18 ..........................................N......N  1
## 19 .........................................N........  1
## 20 ..................................N.N.............  1
## 21 ...............................N.-................  1
## 22 ...............................-..................  1
## 23 ...........................N......................  1
## 24 ...........................-..-...................  1
## 25 ..........................-.......................  1
## 26 .........................N........................  1
## 27 ......................-.........N.................  1
## 28 ......................N...........................  1
## 29 ................N.................................  1
## 30 .............N......................N.............  1
## 31 ..........-.......................................  1
## 32 ........N.........................................  1
## 33 .....N............................................  1
## 34 ..-...........................-...................  1
```

Aligned raw reads may contain missing information in the form of gaps, noted as ‘-’, or indeterminates, noted as ‘N’. `CorrectGapsAndNs` returns the alignment with these positions corrected based on the reference sequence, in this case the dominant haplotype.

```
lstCorrected<-CorrectGapsAndNs(lstCollapsed$hseqs[2:length(lstCollapsed$hseqs)],
                lstCollapsed$hseqs[[1]])
#Add again the most abundant haplotype.
lstCorrected<- c(lstCollapsed$hseqs[1],lstCorrected)
lstCorrected
##   A DNAStringSet instance of length 34
##      width seq                                           names
##  [1]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 1
##  [2]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 2
##  [3]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 3
##  [4]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 4
##  [5]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 5
##  ...   ... ...
## [30]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 30
## [31]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 31
## [32]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 32
## [33]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 33
## [34]    50 TGACGCGCACAGAGTGCTGCT...GGGTTACCCCGTCGTGGTCGC 34
```

After these corrections, some sequences may be duplicated, so it is useful to recollapse the alignment to obtain corrected haplotypes with updated frequencies.

```
lstRecollapsed<-Recollapse(lstCorrected,lstCollapsed$nr)
lstRecollapsed
## $nr
## [1] 100
##
## $seqs
##   A DNAStringSet instance of length 1
##     width seq                                         names
## [1]    50 TGACGCGCACAGAGTGCTGCTA...GGGTTACCCCGTCGTGGTCGC 1
```

3.0.2 Forward and reverse strand haplotype intersection

A key step in error correction with amplicon NGS is selecting haplotypes above a minimum frequency represented in both strands. In the next example we load forward and reverse haplotypes from two separate fasta files.

```
lstFW <- ReadAmplSeqs("../inst/extdata/ToyData_FWReads.fna",type="DNA")
cat("Reads: ",sum(lstFW$nr),", Haplotypes: ",length(lstFW$nr),"\n",sep="")
## Reads: 63736, Haplotypes: 1153
```

```
lstRV <- ReadAmplSeqs("../inst/extdata/ToyData_RVReads.fna",type="DNA")
cat("Reads: ",sum(lstRV$nr),", Haplotypes: ",length(lstRV$nr),"\n",sep="")
## Reads: 65520, Haplotypes: 1162
```

Haplotypes in each strand that do not reach a minimum frequency of 0.1% are then removed, and haplotypes above this frequency and common to both strands are then selected and their frequencies updated by the function `IntersectStrandHpls`.

```
lstI <- IntersectStrandHpls(lstFW$nr,lstFW$hseqs,lstRV$nr,lstRV$hseqs)

cat("FW and Rv total reads:",sum(lstFW$nr)+sum(lstRV$nr),"\n")
cat("FW and Rv reads above thr:",sum(lstI$pFW)+sum(lstI$pRV),"\n")
cat("FW haplotypes above thr:",sum(lstFW$nr/sum(lstFW$nr)>0.001),"\n")
cat("RV haplotypes above thr:",sum(lstRV$nr/sum(lstRV$nr)>0.001),"\n")
cat("\n")
cat("Reads in FW unique haplotypes:",sum(lstI$pFW[lstI$pRV==0]),"\n")
cat("Reads in RV unique haplotypes:",sum(lstI$pRV[lstI$pFW==0]),"\n")
cat("\n")
cat("Reads in common:",sum(lstI$nr),"\n")
cat("Haplotypes in common:",length(lstI$nr),"\n")
## FW and Rv total reads: 129256
## FW and Rv reads above thr: 99982
## FW haplotypes above thr: 35
## RV haplotypes above thr: 35
##
## Reads in FW unique haplotypes: 874
## Reads in RV unique haplotypes: 1047
##
## Reads in common: 98061
## Haplotypes in common: 1
```

3.1 Simulate quasispecies data

Several functions in this package enable simulation of quasispecies data. This is useful for various proposes, such as comparing data and testing diversity indices. The vignette Simulating Quasispecies Composition provides examples of such data simulation.

4 Quasispecies data exploration

Let's load a toy data on which to exemplify different tasks and functions.

```
lst <- ReadAmplSeqs("../inst/extdata/ToyData_10_50_1000.fna",type="DNA")
lst
## $nr
##  [1] 464  62  39  27  37  16  33  54 248  20
##
## $hseqs
##   A DNAStringSet instance of length 10
##      width seq                                          names
##  [1]    50 CACCCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_0_0001|464|46.4
##  [2]    50 CACTCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_1_0001|62|6.2
##  [3]    50 CACCCTGTGACCAGCGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_1_0002|39|3.9
##  [4]    50 CACCCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAC Hpl_1_0003|27|2.7
##  [5]    50 CACTCTGTGACCAGTGTGTGC...TGCCCCGTTAGCATTGACTAT Hpl_2_0001|37|3.7
##  [6]    50 CACTCGGTGACCAGTGTGCGT...TGCCCCGTTGGCATTGACTAT Hpl_4_0001|16|1.6
##  [7]    50 CACTCTGTGACCAGTGTGCGC...TGCCCTGTCGGCATTGACTAT Hpl_5_0001|33|3.3
##  [8]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCATCGACTAT Hpl_8_0001|54|5.4
##  [9]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCATCGACTAC Hpl_9_0001|248|24.8
## [10]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCACCGACTAC Hpl_10_0001|20|2
```

The `ConsSeq` function returns the consensus sequence resulting from an alignment. This function does not consider IUPAC ambiguity codes, and when there is a tie, the consensus nucleotide is decided randomly.

```
ConsSeq(lst$hseqs)
## [1] "CACTCTGTGACCAGTGTGTGCGAGTTTGGTGCCCCGTTGGCATTGACTAT"
```

To visualize the differences between haplotypes that comprise the quasispecies, the `DottedAlignment` function returns a vector of character strings, one for each haplotype, where a dot is shown to represent a conserved site with respect to the dominant haplotype.

```
DottedAlignment(lst$hseqs)
##                                   Hpl_0_0001|464|46.4
## "CACCCTGTGACCAGTGTGTGCGAGTTTGGTGCCCCGTTGGCATTGACTAT"
##                                     Hpl_1_0001|62|6.2
## "...T.............................................."
##                                     Hpl_1_0002|39|3.9
## ".............C...................................."
##                                     Hpl_1_0003|27|2.7
## ".................................................C"
##                                     Hpl_2_0001|37|3.7
## "...T................................A............."
##                                     Hpl_4_0001|16|1.6
## "...T.G...........C.T.............................."
##                                     Hpl_5_0001|33|3.3
## "...T.............C........A......T..C............."
##                                     Hpl_8_0001|54|5.4
## "...T......T.......C........A......T.CC.....C......"
##                                   Hpl_9_0001|248|24.8
## "...T......T.......C........A......T.CC.....C.....C"
##                                      Hpl_10_0001|20|2
## "...T......T.......C........A......T.CC....CC.....C"
```

The output of `ReadAmplSeqs` can be sorted by the number of mutations with regard to the most abundant haplotype using the function `SortByMutations`, in which the first haplotype is the most similar one and the last haplotype is the one with the largest number of mutations. This function also returns a vector with the number of mutations in each haplotype.

```
lstSorted<-SortByMutations(lst$hseqs,lst$nr)
lstSorted
## $bseqs
##   A DNAStringSet instance of length 10
##      width seq                                          names
##  [1]    50 CACCCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_0_0001
##  [2]    50 CACTCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_1_0001
##  [3]    50 CACCCTGTGACCAGCGTGTGC...TGCCCCGTTGGCATTGACTAT Hpl_1_0002
##  [4]    50 CACCCTGTGACCAGTGTGTGC...TGCCCCGTTGGCATTGACTAC Hpl_1_0003
##  [5]    50 CACTCTGTGACCAGTGTGTGC...TGCCCCGTTAGCATTGACTAT Hpl_2_0001
##  [6]    50 CACTCGGTGACCAGTGTGCGT...TGCCCCGTTGGCATTGACTAT Hpl_4_0001
##  [7]    50 CACTCTGTGACCAGTGTGCGC...TGCCCTGTCGGCATTGACTAT Hpl_5_0001
##  [8]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCATCGACTAT Hpl_8_0001
##  [9]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCATCGACTAC Hpl_9_0001
## [10]    50 CACTCTGTGATCAGTGTGCGC...TGCCCTGCCGGCACCGACTAC Hpl_10_0001
##
## $nr
##  [1] 464  62  39  27  37  16  33  54 248  20
##
## $nm
##  [1]  0  1  1  1  2  4  5  8  9 10
```

The frequencies of nucleotides or amino acids at each position can be computed with the function `FreqMat`.

```
FreqMat(lst$hseqs)
##    1  2  3 4  5 6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
## A  0 10  0 0  0 0  0  0  0 10  0  0 10  0  0  0  0  0  0  0  0  0 10  0  0
## C 10  0 10 3 10 0  0  0  0  0  7 10  0  0  1  0  0  0  5  0  9  0  0  0  0
## G  0  0  0 0  0 1 10  0 10  0  0  0  0 10  0 10  0 10  0 10  0 10  0 10  0
## T  0  0  0 7  0 9  0 10  0  0  3  0  0  0  9  0 10  0  5  0  1  0  0  0 10
##   26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## A  0  0  4  0  0  0  0  0  0  0  0  0  0  1  0  0 10  0  0  0 10  0  0 10  0
## C  0  0  0  0  0  0 10 10 10  6  0  3  4  0  0 10  0  1  3  0  0 10  0  0  3
## G  0  0  6 10  0 10  0  0  0  0 10  0  0  9 10  0  0  0  0 10  0  0  0  0  0
## T 10 10  0  0 10  0  0  0  0  4  0  7  6  0  0  0  0  9  7  0  0  0 10  0  7
```

To take into account the abundance of each haplotype when computing the mutation frequency, the haplotype abundances are passed to the function that computes the same matrix, but with the abundances.

```
FreqMat(lst$hseqs,lst$nr)
##      1    2    3   4    5   6    7    8    9   10  11   12   13   14  15
## A    0 1000    0   0    0   0    0    0    0 1000   0    0 1000    0   0
## C 1000    0 1000 530 1000   0    0    0    0    0 678 1000    0    0  39
## G    0    0    0   0    0  16 1000    0 1000    0   0    0    0 1000   0
## T    0    0    0 470    0 984    0 1000    0    0 322    0    0    0 961
##     16   17   18  19   20  21   22   23   24   25   26   27  28   29   30
## A    0    0    0   0    0   0    0 1000    0    0    0    0 355    0    0
## C    0    0    0 371    0 984    0    0    0    0    0    0   0    0    0
## G 1000    0 1000   0 1000   0 1000    0 1000    0    0    0 645 1000    0
## T    0 1000    0 629    0  16    0    0    0 1000 1000 1000   0    0 1000
##     31   32   33   34  35   36  37  38  39   40   41   42  43  44   45   46
## A    0    0    0    0   0    0   0   0  37    0    0 1000   0   0    0 1000
## C    0 1000 1000 1000 645    0 322 355   0    0 1000    0  20 322    0    0
## G 1000    0    0    0   0 1000   0   0 963 1000    0    0   0   0 1000    0
## T    0    0    0    0 355    0 678 645   0    0    0    0 980 678    0    0
##     47   48   49  50
## A    0    0 1000   0
## C 1000    0    0 295
## G    0    0    0   0
## T    0 1000    0 705
```

We may be interested only in mutated positions, so with `MutsTbl` the matrix obtained reports only the frequency of the mutated nucleotide or amino acid per position.

```
MutsTbl(lst$hseqs)
##   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
## A 0 0 0 0 0 0 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4
## C 0 0 0 3 0 0 0 0 0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
## G 0 0 0 0 0 1 0 0 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## T 0 0 0 0 0 0 0 0 0  0  3  0  0  0  0  0  0  0  5  0  1  0  0  0  0  0  0  0
##   29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
## A  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
## C  0  0  0  0  0  0  0  0  3  4  0  0  0  0  1  3  0  0  0  0  0  3
## G  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
## T  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

As can be done with `FreqMat`, with `MutsTbl` the abundances of the haplotypes can be passed to the function to obtain a mutation table with abundance information.

```
MutsTbl(lst$hseqs, lst$nr)
##   1 2 3   4 5  6 7 8 9 10  11 12 13 14 15 16 17 18  19 20 21 22 23 24 25 26
## A 0 0 0   0 0  0 0 0 0  0   0  0  0  0  0  0  0  0   0  0  0  0  0  0  0  0
## C 0 0 0   0 0  0 0 0 0  0   0  0  0  0 39  0  0  0 371  0  0  0  0  0  0  0
## G 0 0 0   0 0 16 0 0 0  0   0  0  0  0  0  0  0  0   0  0  0  0  0  0  0  0
## T 0 0 0 470 0  0 0 0 0  0 322  0  0  0  0  0  0  0   0  0 16  0  0  0  0  0
##    27  28 29 30 31 32 33 34  35 36  37  38 39 40 41 42 43  44 45 46 47 48 49
## A   0 355  0  0  0  0  0  0   0  0   0   0 37  0  0  0  0   0  0  0  0  0  0
## C   0   0  0  0  0  0  0  0   0  0 322 355  0  0  0  0 20 322  0  0  0  0  0
## G   0   0  0  0  0  0  0  0   0  0   0   0  0  0  0  0  0   0  0  0  0  0  0
## T   0   0  0  0  0  0  0  0 355  0   0   0  0  0  0  0  0   0  0  0  0  0  0
##    50
## A   0
## C 295
## G   0
## T   0
```

When the sequences are particularly large, the function `SummaryMuts` computes a table showing the polymorphic positions in the alignment and the frequency of each nucleotide or amino acid observed.

```
SummaryMuts(lst$hseqs,lst$nr,off=0)
##    pos   A   C   G   T
## 4    4   0 530   0 470
## 6    6   0   0  16 984
## 11  11   0 678   0 322
## 15  15   0  39   0 961
## 19  19   0 371   0 629
## 21  21   0 984   0  16
## 28  28 355   0 645   0
## 35  35   0 645   0 355
## 37  37   0 322   0 678
## 38  38   0 355   0 645
## 39  39  37   0 963   0
## 43  43   0  20   0 980
## 44  44   0 322   0 678
## 50  50   0 295   0 705
```

Then, the `PolyDist` function can be used to obtain the fraction of substitutions by polymorphic site in a simpler manner. This function can be used either with or without the vector of abundances.

```
PolyDist(lst$hseqs,lst$nr)
##     4     6    11    15    19    21    28    35    37    38    39    43
## 0.470 0.016 0.322 0.039 0.371 0.016 0.355 0.355 0.322 0.355 0.037 0.020
##    44    50
## 0.322 0.295
PolyDist(lst$hseqs)
##   4   6  11  15  19  21  28  35  37  38  39  43  44  50
## 0.3 0.1 0.3 0.1 0.5 0.1 0.4 0.4 0.3 0.4 0.1 0.1 0.3 0.3
```

To summarize the mutation information and compute the coverage of each mutation, the `ReportVariants` function is used. This function requires a reference sequence, which in some cases could be the dominant haplotype.

```
ReportVariants(lst$hseqs[2:length(lst$hseqs)],lst$hseqs[[1]],lst$nr)
##    WT Pos Var Cov
## 1   C   4   T 879
## 2   T   6   G  37
## 3   C  11   T 335
## 4   T  15   C  62
## 5   T  19   C 388
## 6   C  21   T  37
## 7   G  28   A 351
## 8   C  35   T 351
## 9   T  37   C 335
## 10  T  38   C 351
## 11  G  39   A  27
## 12  T  43   C 248
## 13  T  44   C 335
## 14  T  50   C 341
```

Another way to explore the positions with mutations is by computing a matrix with the information content (IC) of each position using `GetInfProfile`. If the sample is DNA, the maximum IC is 2, whereas when working with amino acids, the maximum is 4.32.

```
GetInfProfile(lst$hseqs,lst$nr)
##        1        2        3        4        5        6        7        8
## 2.000000 2.000000 2.000000 1.002598 2.000000 1.881650 2.000000 2.000000
##        9       10       11       12       13       14       15       16
## 2.000000 2.000000 1.093457 2.000000 2.000000 2.000000 1.762312 2.000000
##       17       18       19       20       21       22       23       24
## 2.000000 2.000000 1.048563 2.000000 1.881650 2.000000 2.000000 2.000000
##       25       26       27       28       29       30       31       32
## 2.000000 2.000000 2.000000 1.061546 2.000000 2.000000 2.000000 2.000000
##       33       34       35       36       37       38       39       40
## 2.000000 2.000000 1.061546 2.000000 1.093457 1.061546 1.771636 2.000000
##       41       42       43       44       45       46       47       48
## 2.000000 2.000000 1.858559 1.093457 2.000000 2.000000 2.000000 2.000000
##       49       50
## 2.000000 1.124907
```

And this can be plotted:

```
dplot <- data.frame(IC=GetInfProfile(lst$hseqs,lst$nr),
                    pos=1:width(lst$hseqs)[1])

ggplot(dplot, aes(x=pos, y=IC)) + geom_point() +
scale_x_continuous(minor_breaks = 1:nrow(dplot), breaks = 1:nrow(dplot)) +
theme(axis.text.x = element_text(angle=45))
```

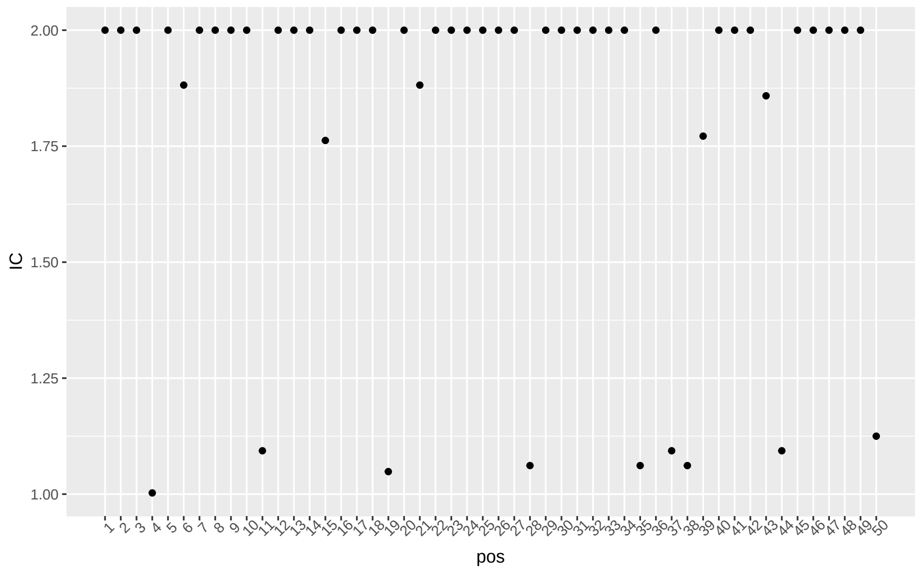

Figure 1: Information content per position in the alignment

5 Quasispecies complexity by biodiversity indices

There is a vignette that deepens with the diversity indices of the quasispecies: Characterizing viral quasispecies is also available in this package.

6 Genotyping

Another interesting procedure is to genotype an unknown sample. To that end, a set of reference sequences is needed for each genotype. A minimum of five well characterized sequences by genotype is suggested. The sets are supposed to be representative of each genotype and will provide an estimate of within genotype variance. Function `DBrule`, used in genotyping, takes into account the distance between the target haplotype and each genotype and the within genotype variability.

The first step is to load the target haplotype to be genotyped with `ReadAmplSeqs`, as is shown:

```
lst2Geno <- ReadAmplSeqs("../inst/extdata/Unknown-Genotype.fna",type="DNA")
hseq <- lst2Geno$hseq[1]
hseq
##   A DNAStringSet instance of length 1
##     width seq                                             names
## [1]   285 CACGTCGCATGGAGACCACCGT...TCAAGCCTCCAAGCTGTGCCT Hpl.0.0001|18|100
```

The reference genotype sequences are then loaded using the same procedure.

```
lstRefs <- ReadAmplSeqs("../inst/extdata/GenotypeStandards_A-H.fas",type="DNA")
RefSeqs <- lstRefs$hseq
{ cat("Number of reference sequences by genotype:\n")
    print(table(substr(names(RefSeqs),1,1)))
}
## Number of reference sequences by genotype:
##
##  A  B  C  D  E  F  G  H
##  9 15 18 15  6  7  5  7
```

Next, the distances between the target haplotype and the reference haplotypes are computed. The matrix of distances between reference haplotypes is stored, in the next code sniped, in dgrp, whereas the distances of the target haplotype to the reference sequences are stored in vector d. The `DBrule` function computes then the most likely genotype based on both, the distances from the target haplotype to the references, and the distances between references of the same genotype.

```
dm <- as.matrix(DNA.dist(c(hseq,RefSeqs),model="K80"))
dgrp <- dm[-1,-1]
d <- dm[1,-1]
grp <- factor(substr(rownames(dgrp),1,1))
hr <- as.integer(grp)
dsc <- DBrule(dgrp,hr,d,levels(grp))
print(dsc)
## $Phi2
##       Phi2.A       Phi2.B       Phi2.C       Phi2.D       Phi2.E
## 0.0063243414 0.0057484677 0.0027527054 0.0006078818 0.0007949051
##       Phi2.F       Phi2.G       Phi2.H
## 0.0188773955 0.0448899802 0.0249913247
##
## $DB.rule
## [1] 4
##
## $Type
## [1] "D"
```

The target sequence has been classified as genotype D, giving the lowest Phi square value.

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.1 LTS
##
## Matrix products: default
## BLAS: /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.7.1
## LAPACK: /usr/lib/lapack/liblapack.so.3.6.0
##
## locale:
##  [1] LC_CTYPE=ca_ES.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=ca_ES.UTF-8        LC_COLLATE=ca_ES.UTF-8
##  [5] LC_MONETARY=ca_ES.UTF-8    LC_MESSAGES=ca_ES.UTF-8
##  [7] LC_PAPER=ca_ES.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=ca_ES.UTF-8 LC_IDENTIFICATION=C
##
```

```
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils     datasets 
## [8] methods   base     
## 
## other attached packages:
##  [1] QSutils_0.99.0      ggplot2_3.0.0       ape_5.1            
##  [4] Biostrings_2.46.0   XVector_0.18.0      IRanges_2.12.0     
##  [7] S4Vectors_0.16.0    BiocGenerics_0.24.0 devtools_1.13.6    
## [10] BiocStyle_2.6.1    
## 
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.17     git2r_0.21.0     bindr_0.1.1      compiler_3.4.4  
##  [5] pillar_1.2.3     plyr_1.8.4       tools_3.4.4      zlibbioc_1.24.0 
##  [9] digest_0.6.15    evaluate_0.10.1  memoise_1.1.0    tibble_1.4.2    
## [13] nlme_3.1-137     gtable_0.2.0     lattice_0.20-35  pkgconfig_2.0.1 
## [17] rlang_0.2.1      psych_1.8.4      yaml_2.2.0       xfun_0.3        
## [21] bindrcpp_0.2.2   withr_2.1.2      stringr_1.3.1    dplyr_0.7.6     
## [25] knitr_1.20       tidyselect_0.2.4 rprojroot_1.3-2  grid_3.4.4      
## [29] glue_1.2.0       R6_2.2.2         foreign_0.8-70   rmarkdown_1.10  
## [33] bookdown_0.7.17  purrr_0.2.5      magrittr_1.5     backports_1.1.2 
## [37] scales_0.5.0     htmltools_0.3.6  mnormt_1.5-5     assertthat_0.2.0
## [41] colorspace_1.3-2 labeling_0.3     stringi_1.2.3    lazyeval_0.2.1  
## [45] munsell_0.5.0
```

References

Domingo, E., J. Sheldon, and C. Perales. 2012. "Viral Quasispecies Evolution." *Microbiology and Molecular Biology Reviews* 76 (2): 159–216. doi:10.1128/MMBR.05023-11.

Gregori, Josep, Celia Perales, Francisco Rodriguez-Frias, Juan I. Esteban, Josep Quer, and Esteban Domingo. 2016. "Viral quasispecies complexity measures." doi:10.1016/j.virol.2016.03.017.

Gregori, Josep, Miquel Salicrú, Esteban Domingo, Alex Sanchez, Juan I. Esteban, Francisco Rodríguez-Frías, and Josep Quer. 2014. "Inference with viral quasispecies diversity indices: Clonal and NGS approaches." *Bioinformatics* 30 (8): 1104–11. doi:10.1093/bioinformatics/btt768.